AMENDMENTS TO THE SPECIFICATION

Please amend paragraph [0003] as follows:

It will be recognized by the skilled artisan that in an application where multiple, repeated connections will be required, a tremendous amount of computing resources can be consumed merely opening connections when required. Yet, particularly in data intensive applications involving repeated remote connections to back end database servers, so much can be the case. Further, it needn't always be the case that a connection can be established reliably. In this regard, when a socket cannot be established between the processes, the attempt can fail leaving little recourse for the client process. Unfortunately, the failure to attain a connection for use by the client process can range from too few available connections to malfunctioning connections.

Please amend paragraph [0005] as follows:

In managing a pool of idle connections, an efficient process consuming little computing overhead can be required to validate that the connections which have not been provisioned recently remain valid and useable on demand. In this regard, it is known to perform placebo transactions using the idle connections to ensure the reliability of the idle connections. Of course, to perform the placebo transaction with respect to any one idle connection necessarily consumes the idle connection such that any attempt to access the idle connection can be blocked. While the liberal use of threads can overcome the blocking action of the placebo transaction, spawning a great many threads for concurrent usage can consume significant computing resources. Thus, in a pool of multiple idle connections, validating each connection using threaded validation

processes can produce the same level of computing overhead that otherwise would exist in consequence of invalid idle connections.

Please amend paragraph [0005] as follows:

FIG. 1 is a schematic illustration of a connection manager configured to manage a connection pool in accordance with the present invention. A connection manager 130 can be coupled to a connection pool 110 and one or more client processes 140. The client processes 140 can be directly coupled to the connection manager 130, consequently ~~in consequence of which~~ individual ones of the client processes 140 can directly request the use of a connection 120A, 120B, 120n disposed within the connection pool 110. Alternatively, the client processes 140 can request the usage of a connection 120A, 120B, 120n intermediately through a server process (not shown) coupled to the connection manager 130.